How to Use Webpack in WordPress

BY HENNER SETYONO / ON 29 NOV 2020 / 14 COMMENTS

#FRONTEND

Does a WordPress developer need to know Webpack?

You might think it is unnecessary and too complex. So I will show you how useful it is for developing a theme.

TABLE OF CONTENTS

1. What is Webpack?

Webpack is a command-line tool to **allow the use of JavaScript modules**. This is a feature that is quite a pain to do without Webpack.

For example, we have `app.js` that needs 2 libraries: `js-datepicker.js` and `swiper.js`. In the old way, you would enqueue them separately resulting in this output:

```html
<script src="/wp-content/themes/mytheme/js/js-datepicker.js">
<script src="/wp-content/themes/mytheme/js/swiper.js">
<script src="/wp-content/themes/mytheme/js/app.js">
```
HTML

With Webpack, we can use `import` to embed the libraries inside `app.js`. All the codes are now combined into one file:

```
js/app.js

import datepicker from './js-datepicker.js';
import swiper from './swiper.js';

// do something
JS
```

```
<!-- Now only need to enqueue 1 file -->

<script src="/wp-content/themes/mytheme/dist/app.js">
HTML
```

This seems like a **minor benefit** but it will become more apparent when you have a lot of files. Moreover, Webpack has many other utilities such as:

- Minify your code
- Compiling Sass into CSS
- Auto reload your CSS as you save
- Allow use of advanced syntaxes for your Vue or React code.
- etc

In this tutorial, we will learn to do the first three of that list.

2. How to Install / Setup Webpack?

1. Download and install Node JS. Pick the recommended version.
2. Create `package.json` and `webpack.config.js` files in your theme and paste in the code below:

```
wp-content/my-theme/package.json

{
  "name": "my-theme",
  "private": true,
  "dependencies": {},
  "devDependencies": {
    "browser-sync": "^2.26.12",
    "browser-sync-webpack-plugin": "^2.2.2",
    "css-loader": "^4.3.0",
    "file-loader": "^6.1.0",
    "mini-css-extract-plugin": "^0.11.2",
    "sass": "^1.53.0",
    "sass-loader": "^10.0.2",
```

```json
    "url-loader": "^4.1.0",
    "webpack": "^4.44.1",
    "webpack-cli": "^3.3.12"
  },
  "scripts": {
    "build": "webpack --mode production",
    "dev": "webpack --mode development --watch"
  }
}
```
JSON

wp-content/my-theme/webpack.config.js

```js
const MiniCssExtractPlugin = require('mini-css-extract-plugin');
const BrowserSyncPlugin = require('browser-sync-webpack-plugin');
var path = require('path');

// change these variables to fit your project
const jsPath= './js';
const cssPath = './css';
const outputPath = 'dist';
const localDomain = 'http://mysite.local';
const entryPoints = {
  // 'app' is the output name, people commonly use 'bundle'
  // you can have more than 1 entry point
  'app': jsPath + '/app.js',
};

module.exports = {
  entry: entryPoints,
  output: {
    path: path.resolve(__dirname, outputPath),
    filename: '[name].js',
  },
  plugins: [
    new MiniCssExtractPlugin({
      filename: '[name].css',
    }),

    // Uncomment this if you want to use CSS Live reload
    /*
    new BrowserSyncPlugin({
      proxy: localDomain,
      files: [ outputPath + '/*.css' ],
      injectCss: true,
    }, { reload: false, }),
    */
  ],
  module: {
    rules: [
      {
        test: /\.s?[c]ss$/i,
        use: [
          MiniCssExtractPlugin.loader,
          'css-loader',
          'sass-loader',
        ],
      },
      {
        test: /\.sass$/i,
        use: [
          MiniCssExtractPlugin.loader,
          'css-loader',
          {
            loader: 'sass-loader',
            options: {
              sassOptions: { indentedSyntax: true },
            },
          },
        ],
      },
      {
        test: /\.(jpg|jpeg|png|gif|woff|woff2|eot|ttf|svg)$/i,
        use: 'url-loader?limit=1024',
      },
    ]
  },
};
```

JS

The code above assumes your theme's structure looks like below. If it's different, change the variables in Line 5 accordingly:

```
/
├── css/
│   ├── app.css
├── js/
│   ├── app.js
├── dist/
├── index.php
├── page.php
├── ...
```

3. Open command-prompt / terminal in your theme and **run the command below**. For Windows users, you might see an error saying unable to remove the directory. Simply rerun the command to solve that.

```
npm install
```

4. To start compiling, run the command below. Every time you save your JS, it will automatically re-compile.

```
npm run dev
```

5. Before launching, you need to **minify the code**. To do this, you need to quit the `dev` command, then run:

```
npm run build
```

3. Compiling CSS or Sass

By default, Webpack only supports JS files. But if you followed our setup above, it has extra packages to handle CSS or Sass.

There are 2 ways of compiling them. The **first way** is to directly import them in your JS:

```
js/app.js

import '../css/style.sass'

// ...
                                                                 JS
```

The **second way** is to add a new entry point in your config:

```
webpack.config.js

// ...

const entryPoints = {
  'app': jsPath + '/app.js',
  'style': cssPath + '/style.sass',
};

// ...
                                                                 JS
```

The first way makes more sense if the CSS is related to the script.

> **Note**: If you get this error: `ENOENT no such file or directory, scandir`, you can fix it by running the command `npm rebuild node-sass`.

4. CSS Live Reload

Whenever you save your CSS, Webpack will **automatically reloads it**.

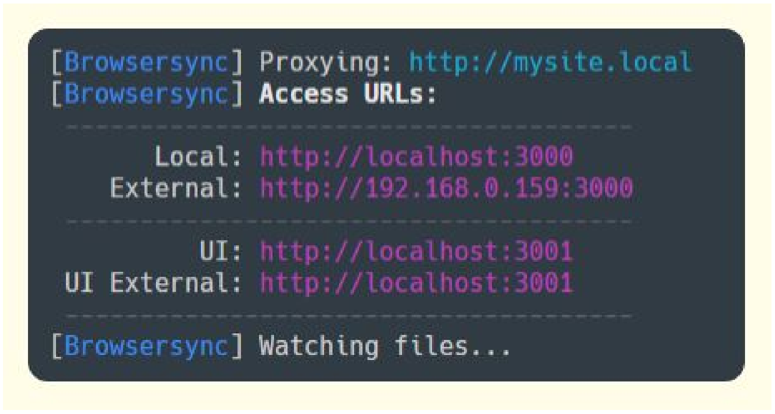
You are probably using other tools to do this, but it’s nice to have everything in one place.

We have included the packages to handle this in our current setup. Simply do these 2 steps in your `webpack.config.js` to enable it:

- Update `localDomain` variable to be the local domain you use for your project.
- Uncomment the `new BrowserSyncPlugin` block.

To launch it, use the same command:

```
npm run dev
```



```
[Browsersync] Proxying: http://mysite.local
[Browsersync] Access URLs:
 -----------------------------------------
       Local: http://localhost:3000
    External: http://192.168.0.159:3000
 -----------------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 -----------------------------------------
[Browsersync] Watching files...
```

Command Line of Browser Sync

If you open your site in `localhost:3000`, any updated CSS will be reloaded.

Also, do you notice the **External URL**? You can open that **on your mobile phone** to view your local site! As long as both your computer and phone are under the same Wi-Fi.

> **Note**: Depending on your router setup, the External URL might not work. I'm not familiar with networking stuff, so I don't know the fix. Let me know in the comment if you found one.

5. Getting JS Library from NPM

Most popular JS libraries are available in NPM. So you don't need to save a local copy in your theme.

Simply **Google it** to know whether it's available as NPM or not.

In our example, `js-datepicker` and `swiper` is available as NPM. So we can install them by running these 2 commands (in your theme directory):

```
npm install --save js-datepicker
npm install --save swiper
```

Now, we update their import references from this:

```
js/app.js (old)

import datepicker from './js-datepicker.js';
import swiper from './swiper.js';
                                                    JS
```

Into directly using the package's name:

```
js/app.js (new)

import datepicker from 'js-datepicker';
import swiper from 'swiper';
                                                    JS
```

. . .

Conclusion

I hope this convinces you to start using Webpack in your project. Unless you're doing minimal customization to a theme, this will serve you well.

There are limitless possibilities with what you can do with Webpack. I believe this tutorial is good enough for you to be able to experiment further.

> Leave a comment below if you have any question regarding Webpack. I'm not a pro at it, but will try my best in answering yours.

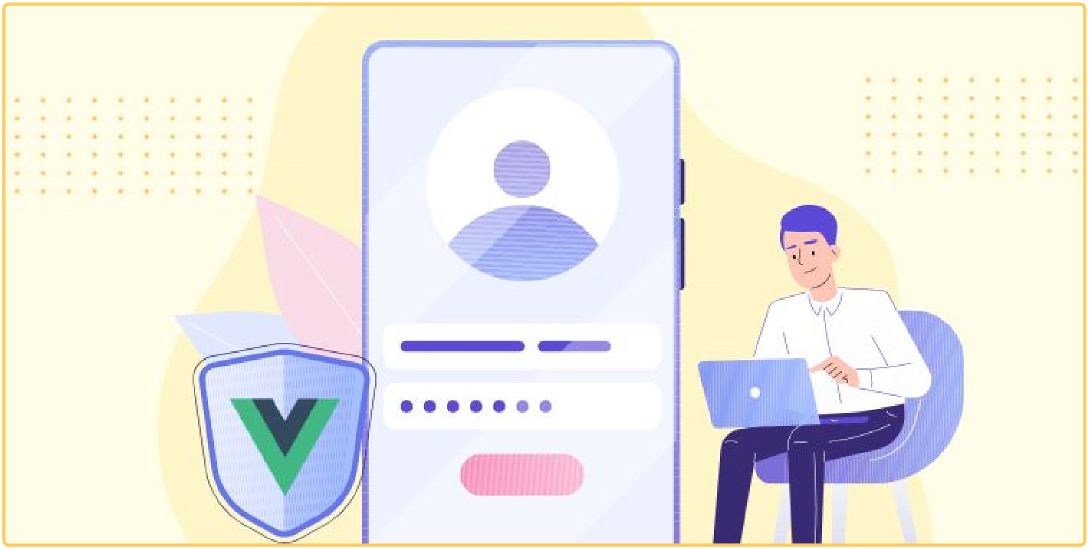
# javascript # webpack

Written by

Henner Setyono

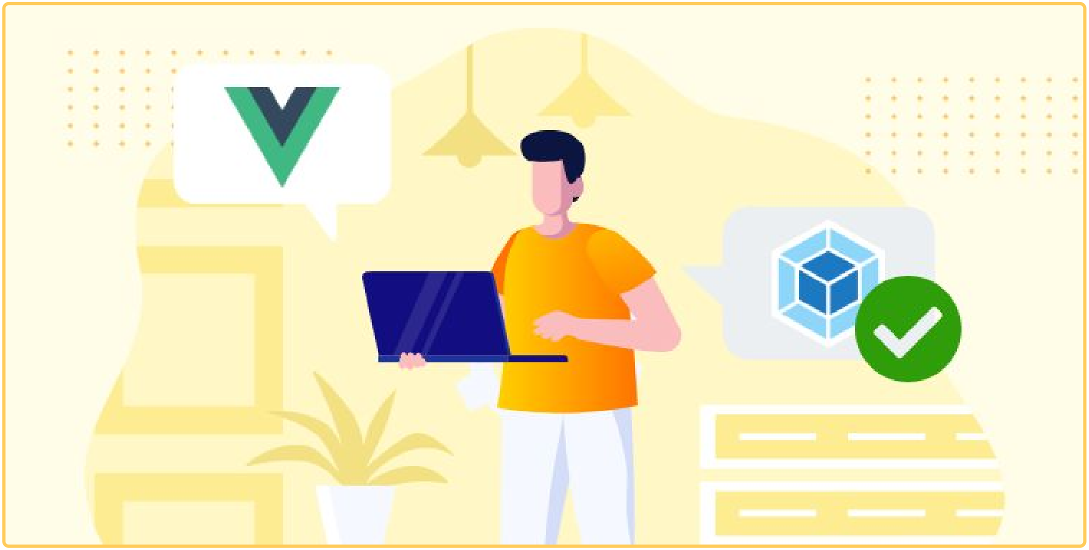
A web developer who mainly build custom theme for WordPress since 2013. Young enough to never had to deal with using Table as layout.

Related Posts



WordPress Login with Vue JS (Authentication API)

On 29 Apr 2022 / 5 Comments



How to Use Vue in WordPress (with Webpack)

On 30 Jan 2022

Should You Use jQuery in 2021? (Well Yes, but No)

On 13 Jan 2021 / 2 Comments

Leave a Reply

Name *	Email *	Website

B *I* </> { } Block

☐ Save my name, email, and website in this browser for the next time I comment.

☐ Notify me of follow-up comments by email.

Post Comment

14 Comments

Jean-Michel

JAN 28, 2021 / 4:32 AM REPLY

Very clear, simple and efficient. I had to adjust the output so it doesn't end up in the dist directory and it's working fine. The live reload is a nice cherry on top! Thank you!

Lukasz

FEB 24, 2021 / 9:12 PM REPLY

Thank you, this is a great and simple explanation! I actually want to try using webpack now, I read countless other articles and thought it would be too complicated.

Peter Liu

JUN 16, 2021 / 12:41 AM REPLY

Thank you for this tutorial! Very clear and straightforward. It did exactly what I needed it to do.

Lew

SEP 12, 2021 / 5:49 PM REPLY

Love this guide, use it all the time. One question though. I use your package.json dependencies list which has approx. 15 dependencies list. However once initialised I have LOADS (hundreds) of node modules in my node_modules folder. Is this normal? why is this? are they needed?

Thanks!

> **Henner Setyono**
>
> SEP 12, 2021 / 9:27 PM REPLY
>
> Hi Lew,
>
> Yes, each dependencies also dependant on something else, so you need to download everything.
>
> Don't worry about the node_modules folder as you don't need to upload that to the live server.
>
> **Rafael Cavalcante**
>
> NOV 11, 2021 / 2:49 AM REPLY

Lew, don't forget to add the node_modules folder to your git ignore file (.gitignore)

Dude

SEP 13, 2021 / 1:25 AM REPLY

What about cache busting?

Henner Setyono

SEP 13, 2021 / 9:14 AM REPLY

Hi Michael,

I do cache busting natively in WordPress during the enqueue process. I usually has a constant with the Theme version like this:

```
$THEME = wp_get_theme();
define( 'THEME_VERSION', $THEME->get( 'Version' ) );
```

Then when enqueuing, you do:

```
add_action( 'wp_enqueue_scripts', 'my_public_assets', 99 );

function my_public_assets() {
  $dir = get_template_directory_uri() . '/dist';
  wp_enqueue_style( 'my-app', $dir . '/app.css', [], THEME_VERSION );
}
```

Unfortunately, this means we got to increment the Theme version in `style.css` everytime there is changes. There is definitely a more automated way for this.

Mads

OCT 6, 2021 / 6:13 PM REPLY

I do the following to set the version based on the css filetime. It's not as good as doing a content based version, but it works for my scenario.

```
$css_ver = date("ymd-Gis", filemtime( get_stylesheet_directory() . '/app.
wp_enqueue_style( 'my-app', get_stylesheet_directory_uri() . '/app.css',
```

Henner Setyono

OCT 6, 2021 / 7:28 PM REPLY

Thanks for sharing! This sounds like a good solution. Never thought of doing this before.

Elena

MAR 26, 2022 / 5:56 PM REPLY

Hey, thank you for nice explanation, may be you could help me to understand how i need to act - i am trying to bundle files in plugin i developing and installed node files into plugins folder, also in the same folder i placed file with config and package.json - the problem when i run command build in command line i am getting an error that it does not see node files. thank you

Henner Setyono

MAR 26, 2022 / 6:09 PM REPLY

Hi Elena, it seems you haven't installed the packages. Run the command `npm install` first.

The package.json needs to be inside that specific plugin's folder too

---

saranasr

AUG 25, 2022 / 10:36 PM REPLY

Very clear, Simple And Great , Can we use css modules too in our wordpress project ? i tried to edit webpack config but i have no luck

Henner Setyono

AUG 26, 2022 / 10:16 AM REPLY

Hi, My current webpack config supports CSS modules. It will be outputted as CSS file with the same name as the JS. Simply change the directory and file name to fit your project

Design by Pixel Studio

About wpTips.dev

wpTips is a non-profit blog focused on **advanced** WordPress tutorial in a concise and beautiful way.

We want to show the world the full potential of our beloved CMS.

Join Our Community

We are a team of enthusiasts who enjoy sharing our knowledge.

Let us know at hello@wptips.dev if you are interested to write some tips!